**Radio Shack®**

# Model
# 4
# Quick
# Reference
# Guide

# Contents

Make sure all floppy disk drives are empty and all equipment is off.

1. Turn on all peripheral equipment (such as a printer), except the hard disk.
2. Hard disk users: Turn on the primary hard disk drive.
3. Turn on the computer.
4. Insert a system diskette into Drive 0 and close the drive door. TRSDOS displays its start-up message.
5. TRSDOS prompts you for the date. Enter it in the *mm/dd/yy* format.
6. The following system prompt will appear on your screen:

   TRSDOS Ready

   Now, you can type in a TRSDOS command.
7. To start BASIC, type:

   BASIC (ENTER)

   and you see the BASIC prompt:

   Ready

   Now, you can type in a BASIC command.

This Quick Reference Guide is divided into two sections: TRSDOS and BASIC.

Information which is non-shaded (like this) pertains to:

- TRSDOS intermediate commands and utilities
- BASIC statements

Information which is shaded *like this* pertains to:

- TRSDOS advanced commands and utilities
- BASIC functions

Those parts of the command line that you must enter are
**highlighted**. Information that is upper-case should be
typed in exactly as is. Information that is lower-case
represents a value that you supply. Information within
brackets is optional.

**APPEND source** [TO] **destination** [(ECHO,STRIP)]
  Adds one disk file onto the end of another.

**ATTRIB file** (USER = "password",OWNER = "password",
  PROT = level, VIS,INV)
  Changes the protection of a file. The *level* can be
  EXEC, READ, UPDATE, WRITE, RENAME, REMOVE,
  or FULL.
    ATTRIB CUSTFILE/DAT:1 (USER="",
    OWNER="BOSSMAN",PROT=READ,VIS)

**ATTRIB** [:drive] **(LOCK,UNLOCK,MPW = "password",
  NAME = "disk name", PW = ["password"])**
  Changes the protection of files on a drive.
    ATTRIB :1 (NAME="DATA",PW="SECRET",
    MPW="BOSSMAN")

**AUTO** [:drive, ?:drive, = :drive] [*] [command line]
  Stores a *command line* for automatic execution each
  time TRSDOS starts up. (AUTO by itself deletes the
  current AUTO command line.)
    AUTO BASIC    AUTO *DO INIT/JCL:1

**BACKUP** [file]:**source drive** [TO] :**destination drive**
[(MPW = "password",SYS, INV, MOD, QUERY = YES, OLD,
  NEW, X, DATE = "date")]
  Duplicates a system or data diskette. (*file* can be a
  partial name.)
    BACKUP :0 :1
    BACKUP (MOD, QUERY, MPW="SECRET")

**BOOT** [(CLEAR), (ENTER), (D)]
  Resets the system.
    BOOT

**BUILD file** [(HEX,APPEND)]
  Creates an input file for JCL, KSM, and other TRSDOS
  commands.
    BUILD MYPROGA/FIX:0
    BUILD DISPLAY/BLD (HEX)

**CAT** [-] [file] [:] [drive 1] [-] [:] [drive 2]
  [(ALL, INV, MOD, NON, PRT, SYS, DATE, SORT = NO)]
  It works same as DIR except ALL parameter is OFF.
    CAT :1 -

CLICK/FLT
**SET device** [TO] **CLICK/FLT** [(CHAR = number)]
  **FILTER *KI device**
  Establishes the key-click filter.

**CLS**
  Clears screen and homes cursor.
    CLS

COM/DVR
**SET ∗CL** [TO] **COM/DVR**
  Prepares the Communications Line (∗CL) for use.

**COMM device**[(XLATES = X'aabb', XLATER = X'aabb',
  XON = X'cc', XOFF = X'cc', NULL = OFF)]
  Lets two computers communicate via a device.

**CONV** [file]:**source drive** [:destination drive] [(VIS,
  INV, SYS, NEW, OLD, QUERY = NO, DIR)]
  Converts files from a TRSDOS 1.3 (Model III) diskette
  onto a TRSDOS Version 6 formatted diskette. (*file* can
  be a partial name.)
    CONV :1 :0 (VIS,Q=NO)

**COPY source** [TO] **destination** [(LRL = nnn,
  CLONE = NO, ECHO,X)]
  Copies data from one file or device to another.
    COPY TEST/DAT TO :1
    COPY ∗KI TO ∗PR (ECHO)

**CREATE file** [(LRL = number, REC = number,
  SIZE = number)]
  Creates a file and reserves space on the disk for future
  use.
    CREATE INVENT/DAT (SIZE=20)

**DATE** [mm/dd/yy]
  Sets or displays the current date.
    DATE 08/09/82    DATE

**DEBUG** [(ON, OFF)] [(EXT)]
  Sets up the debug monitor for testing and debugging
  machine-language programs.
    DEBUG    DEBUG (OFF)

**DEVICE** [(D = NO, B = YES, S = NO, P = YES)]
  Displays the current status of each drive and the
  options in use.
    DEVICE    DEVICE (B=YES)

**DIR** [partspec|-partspec] [:] [drive 1] [-] [:] [drive 2]
  [(ALL = OFF,INV,MOD,NON, PRT, SYS, DATE,
  DATE = "date", SORT = NO)]
  Lists the directory of a drive or file. (*file* can be a partial
  name.)
    DIR :1    DIR (DATE="10/01/81-")

**DO** [$, =,*] **file** [(@label,parm = value)] [:]
Compiles and executes a DO file.
```
DO DRIVE/JCL     DO = DRIVE/JCL
```

**DUMP file (START = address, END = address,**
**TRA = address, ASCII, ETX = value)**
Copies an area of memory to a disk file.
```
DUMP ROUTINE/CMD
(START=X'7000',END=X'8000',
TRA=X'7000')
```

**FILTER device** [USING] **phantom device**
Filters data to or from a device, using a filter program.
```
FILTER *PR USING *DU
```

FLOPPY/DCT
**SYSTEM (DRIVE = d, [DISABLE,] DRIVER = "FLOPPY")**
Lets you define a logical drive as a floppy drive.

**FORMAT :drive (ABS, NAME = "disk name",**
**MPW = "password", SDEN, DDEN, CYL = number,**
QUERY = NO, STEP = number, DIR = number,
SIDES = number)
Formats a blank or old disk for use.
```
FORMAT
FORMAT :1 (NAME="DATA3",
MPW="SECRET")
```

**FORMS** [(DEFAULT, ADDLF, CHARS = number, FFHARD,
INDENT = number, LINES = number,
MARGIN = number, PAGE = number, QUERY, TAB,
XLATE = X'aabb')]
Sets up printer options.
```
FORMS (MARGIN=10,CHARS=80,INDENT=6)
```

FORMS/FLT
**SET *PF** [TO] **FORMS/FLT**
**FILTER *PR *PF**
Prepares the Printer Filter (*PF) for use.

**FREE** [:drive] [(PRT)]
Lists free space and number of files on each disk; if
*drive* is specified, displays space map of that disk.
```
FREE    FREE :0 (PRT)
```

**HELP** [filespec] [*] [keyword] [(parameters = P,V,R,S)]
Displays information about TRSDOS keywords.
```
HELP DOS
HELP * ATTRIB
```

JOBLOG
  **ROUTE \*JL** [TO] **file**
  **ROUTE \*JL** [TO] **device**
  Establishes the Joblog device (\*JL), which sends
  certain information to a file or device.
    ROUTE \*JL TO LISTER/JBL
    ROUTE \*JL TO \*PR

KSM/FLT
  **SET device KSM/FLT** [USING] **file**
  [(ENTER = value)]
  **FILTER \*KI device**
    SET \*DU KSM/FLT USING ROUTINE/KSM
    FILTER \*KI \*DU

**LIB**
  Displays library commands.
    LIB

**LINK device1** [TO] **device2**
  Links two logical devices.
    LINK \*DO \*PR

**LIST file** [(ASCII8,NUM,HEX, TAB = number, PRT,
  LINE = number, REC = number, LRL = number)]
  Lists contents of a file to the display or printer.
    LIST TESTFILE:0
    LIST MONITOR/CMD (PRT)

**LOAD** [(X)] **file**
  Loads a program file into memory.
    LOAD STATUS/CMD
    LOAD (X) PROGRAM/CIM

**LOG :drive**
  Logs in a minimum or full system disk, or a double-
  sided disk in Drive 0.
    LOG :1

MEMDISK/DCT
  **SYSTEM (DRIVE = drive, DRIVER = "MEMDISK")**
  Adds to the system a pseudo floppy drive which keeps
  its files in memory.
    SYSTEM (DRIVE=2,DRIVER="MEMDISK")

**MEMORY** [(CLEAR = value, HIGH = address,
  LOW = address, ADD = address, WORD = word,
  BYTE = byte, GO = address)]
  Reserves a portion of memory, sets or displays current
  HIGH\$ and LOW\$, modifies a memory address, or
  jumps to a specified memory location.
    MEMORY

**PATCH file (patch commands)**
  Changes the contents of a disk file.
```
PATCH MONITOR/CMD (X'E100'=C3 66 00)
```

**PATCH file1** USING **file2** [(YANK,REMOVE)]
  Makes changes contained in *file2* to *file1*.
```
PATCH BACKUP/CMD:0 USING SPECIAL/FIX
```

**PURGE** [file]:**drive** [(QUERY = NO, MPW = "password",
  INV,SYS, DATE = "date")]
  Deletes files. (*file* can be a partial name.)
```
PURGE :0 (MPW="SECRET")
PURGE /BAS:1 (Q=N)
```

**REMOVE file** [file] ...
  Deletes files from the directory.
```
REMOVE ALPHA/DAT:0 BREAKER/DAT:0
```

**REMOVE device** [device] ...
  Removes devices from the device table.
```
REMOVE *LU
```

**RENAME file1** [TO] **file2**
**RENAME device1** [TO] **device2**
  Changes the name of a file or device.
```
RENAME TEXT/DAT:0 TO OLD/DAT
RENAME *UD TO *TX
```

**REPAIR :drive**
  Updates system information on disks which were
  formatted under Model I TRSDOS.
```
REPAIR :1
```

**RESET device**
**RESET file**
  Returns a device to its original start-up condition.
  Closes an open file.
```
RESET *PR    RESET PRINTER/DAT
```

**ROUTE device1** [TO] **device2**
**ROUTE device** [TO] **file** [(REWIND)]
**ROUTE device (NIL)**
  Routes a device to another device, to a disk file, or to
  nothing (NIL).
```
ROUTE *PR *DO
ROUTE *PR TO PRINTER/DAT
```

[RUN] [(X)] **file** [(command text)]
  Loads and executes a program. *command text* is
  optional values the program may require.
```
RUN CONTROL/CMD    CONTROL/CMD
```

9

**SET device** [TO] **driver file** [(parameters)]
   Assigns a driver program to a device. *parameters* are
   optional values the driver program may require.
   ```
   SET *SP TO SERIAL/DRV
   ```

**SET phantom device** [TO] **filter file** [USING]
   [parameters]
   Assigns a filter program to a phantom device.
   *parameters* are optional values the filter program may
   require.
   ```
   SET *LC TO TRAP/FLT
   ```

**SETCOM** [(DEFAULT, BAUD = number, WORD =
   number, STOP = number, PARITY = switch, QUERY,
   BREAK = value, EVEN,ODD)]
   Sets up RS-232C communications or display status.
   ```
   SETCOM (BAUD=300,WORD=8,STOP=1,
   PARITY=OFF)
   ```

**SETKI** [(DEFAULT, RATE = number, WAIT = number,
   QUERY)]
   Sets keyboard repeat values. (SETKI by itself displays
   current values.)
   ```
   SETKI (DELAY=15)
   ```

**SPOOL** [device] [TO] [file] (NO, **MEM = number,
   BANK = number, DISK = number,** PAUSE,
   RESUME,CLEAR)
   Establishes an output buffer for a device.
   ```
   SPOOL *PR TO TEXTFILE:0
   (MEM=5,DISK=15)          SPOOL *PR (NO)
   ```

**SYSGEN** [(switch)] [(DRIVE = drive)]
   Stores current system options in a file (CONFIG/SYS)
   on *drive*. If *switch* is NO, the configuration file is
   removed.
   ```
   SYSGEN (YES) (DRIVE=4)      SYSGEN (NO)
   ```

**SYSTEM (parameters)**
   Selects certain options of your TRSDOS system. In
   the following SYSTEM commands, *switch* is YES or
   NO.
   **SYSTEM (ALIVE**[ = *switch*]**)**
      Displays a moving character when task processor is
      running.
   **SYSTEM (BLINK** = *switch***)**
   **SYSTEM (BLINK** = *number***)**
   **SYSTEM (BLINK,**[LARGE|SMALL]**)**
      Control the cursor character.
   **SYSTEM (BREAK**[ = *switch*]**)**
      Enables or disables BREAK key.

**SYSTEM (BSTEP** = *number***)**
   Establishes default bootstrap step rate used by
   FORMAT.
**SYSTEM (DATE**[ = *switch*]**)**
   Turns on or off the start-up date prompt.
**SYSTEM (DRIVE** = *drive*, [CYL = *number*,
   DELAY = NO/YES, DISABLE, ENABLE,
   DRIVER = "*file*", WP = *switch*]**)**
   Sets parameters for *drive*.
**SYSTEM (FAST)**
   Sets system to run at 4 MHz, Model 4 speed.
**SYSTEM (GRAPHIC** [ = *switch*]**)**
   Informs system that printer has capability of
   reproducing TRS-80 graphics during screen-print.
**SYSTEM (RESTORE** [ = *switch*]**)**
   Enables/disables the restoring of all drives to Track
   0 at start up.
**SYSTEM (SLOW)**
   Sets system at 2 MHz, Model III speed.
**SYSTEM (SMOOTH** [ = *switch*]**)**
   Allows smooth disk access.
**SYSTEM (SYSRES** = *number***)**
   Adds TRSDOS system overlays into high memory.
**SYSTEM (SYSTEM** = *drive***)**
   Assigns *drive* as system drive.
**SYSTEM (TIME**[ = *switch*]**)**
   Turns on or off the start-up time prompt.
**SYSTEM (TRACE**[ = *switch*]**)**
   Displays contents of Program Counter.
**SYSTEM (TYPE**[ = *switch*]**)**
   Turns on or off the KI/DVR type-ahead feature.

TAPE100
   **TAPE100 file1** [TO] **file2 (READ,WRITE)**
   Reads a Model 100 cassette tape file and writes it to a
   disk file, or reads a disk file and writes it to cassette
   tape.
   ```
   TAPE100 PRNTER TO PRINT/DAT:0 (READ)
   ```

**TIME** [hh:mm:ss] [(CLOCK = YES/NO)]
   Sets the time or displays current time.
   ```
   TIME     TIME 12:29:34
   ```

**TOF**
   Sends top-of-form to printer.
   ```
   TOF
   ```

**VERIFY** [(switch)]
   Sets VERIFY function on or off.
   ```
   VERIFY (YES)     VERIFY (NO)
   ```

# Notes:

# Error Messages

| Number | Message | Explanation/Action |
|--------|---------|--------------------|
| 7<br>X'07' | Attempted to read locked/deleted data record | Check for error in program |
| 6<br>X'06' | Attempted to read system data record | Check for error in program |
| 5<br>X'05' | Data record not found during read | Try again; use another disk; reformat old disk |
| 13<br>X'0D' | Data record not found during write | Try again; use another disk |
| 39<br>X'27' | Device in use | Reset device in use before REMOVEing it |
| 8<br>X'08' | Device not available | Check device specification; make sure peripheral is ready |
| 30<br>X'1E' | Directory full — can't extend file | Copy files to new disk |
| 17<br>X'11' | Directory read error | Try another drive or disk |
| 18<br>X'12' | Directory write error | Try another disk |
| 27<br>X'1B' | Disk space full | Write file to a disk with more available space |
| 28<br>X'1C' | End of file encountered | Check for error in program |
| 63 | Extended error | Error code is in HL register |
| 25<br>X'19' | File access denied | Use correct password; use no password for unprotected file |
| 41<br>X'29' | File already open | Use RESET to close the file |
| 24<br>X'18' | File not in directory | Check spelling of filespec |
| 38<br>X'26' | File not open | Open file before access |
| 20<br>X'14' | GAT read error | Try another drive |
| 21<br>X'15' | GAT write error | Try another drive or disk |
| 22<br>X'16' | HIT read error | Try another drive |

| 23 X'17' | HIT write error | Try another drive or disk |
|---|---|---|
| 37 X'25' | Illegal access attempted to protected file | OWNER password is required for the requested access |
| 32 X'20' | Illegal drive number | Drive is not in system or not ready for access |
| 19 X'13' | Illegal file name | Use proper filespec syntax |
| 16 X'10' | Illegal logical file number | Check for error in program |
| 34 X'22' | Load file format error | Attempt was made to load a non-program file |
| 3 X'03' | Lost data during read | Try another drive or disk |
| 11 X'0B' | Lost data during write | Try another drive or disk |
| 42 X'2A' | LRL open fault | COPY file to another file that has the specified LRL |
| 33 X'21' | No device space available | REMOVE non-system devices to provide more space |
| 26 X'1A' | No directory space available | Use a different disk or REMOVE unwanted files |
| 0 X'00' | No error | Check for error in program |
| 1 X'01' | Parity error during header read | Try another drive or disk |
| 9 X'09' | Parity error during header write | Try another drive or disk |
| 4 X'04' | Parity error during read | Try another drive or disk |
| 12 X'0C' | Parity error during write | Try another drive or disk |
| 31 X'1F' | Program not found | Check spelling of filespec; check for proper disk in drive |
| 40 X'28' | Protected system device | System devices cannot be REMOVEd |
| 29 X'1D' | Record number out of range | Provide correct record number or try another copy of the file |

| | | |
|---|---|---|
| 2<br>X'02' | Seek error during read | Set step rate with SYSTEM command or try another drive or disk |
| 10<br>X'0A' | Seek error during write | Set step rate with SYSTEM command or try another drive or disk |
| — | Unknown error code | Check for error in program |
| 14<br>X'0E' | Write fault on disk drive | Try another disk or drive |
| 15<br>X'0F' | Write protected disk | Remove write-protect tab or write enable disk using SYSTEM command |
| 43<br>X'2B' | SVC parameter error | Check program to see if correct parameter is passed |
| 44<br>X'2C' | Parameter error | Check for spelling or value or abbreviation of the parameter |
| 63<br>X'3F' | Extended error | Check for error in program |

# BASIC Statements and Functions

**Terms:**

integer:
a whole number from $-32768$ to $32767$
string:
a sequence of characters which is to be taken
verbatim
dummy number or dummy string:
a number or string used in an expression to meet
syntactic requirements, but whose value is
insignificant.

**ABS** (*number*)
Computes the absolute value of *number*.
```
Y = ABS(X)
```

**ASC** (*string*)
Returns the ASCII code for the first character of *string*.
```
PRINT ASC("A")
```

**ATN** (*number*)
Computes the arctangent of *number*; returns the value
in radians.
```
Y = ATN(X/3)
```

**AUTO** [*line number*] [*,increment*]
Automatically generates line numbers every time you
press (ENTER). AUTO begins numbering at *line number*
and displays the next line using *increment*.
```
AUTO    AUTO 1000, 100    AUTO , 5
```

**CALL** *address* [*(parameter list)*]
Transfers program control to an assembly-language
subroutine stored at *address*. The *parameter list*
contains the values to be passed to the external
subroutine.

**CDBL** (*number*)
Converts *number* to double precision.
```
Y# = CDBL(N*3)
```

**CHR$** (*code*)
Returns the corresponding character of the ASCII or
control *code*.
```
PRINT CHR$(35)
```

**CHAIN** [MERGE] *"filespec"* [,*line*] [,ALL] [,DELETE
  *line - line*]
  Loads a BASIC program named *filespec*, chains it to a
  "main" program, and begins running it. The *line* is the
  first line to be run in the CHAINed program. The ALL
  option passes every variable in the main program to
  the CHAINed program. The MERGE option "overlays"
  the lines of *filespec* with the main program. The
  DELETE option erases lines in the overlay so that you
  can MERGE in a new overlay.

**CINT** (*number*)
  Converts *number* to integer representation.
```
PRINT CINT(17.65)
```

**CLEAR** [,*memory location*] [,*stack space*]
  Clears the value of all variables and closes all open
  files. Optionally, it also sets the highest *memory
  location* for BASIC and the amount of *stack space*.
```
CLEAR      CLEAR ,75     CLEAR ,61000,200
```

**CLOSE** [*buffer,...*]
  Closes access to a file. The *buffer* number (the same
  used to OPEN the file) may be from 1 to 15.
```
CLOSE 1, 2, 8
CLOSE FIRST% + COUNT%
```

**CLS**
  Clears the screen.
```
CLS
```

**COMMON** *variable,...*
  Passes one or more variables to a CHAINed program.
```
100 COMMON A, B, C, D(), G$
110 CHAIN "PROG3", 10
```

**CONT**
  Resumes execution of a program when it has been
  stopped by the (BREAK) key or by a STOP or an END
  statement in the program.
```
CONT
```

**COS** (*number*)
  Computes the cosine of *number*.
```
Y = COS(X * .01745329)
```

**CSNG** (*number*)
  Converts *number* to single precision.
```
CSNG(.1453885509)
```

**CVD**(*8-byte string*)
  Restores the string value to a numeric value.
```
A# = CVD (GROSSPAY$)
```

**CVI** (*2-byte string*)
Restores the string value to a numeric value.

**CVS** (*4-byte string*)
Restores the string value to a numeric value.

**DATA** *constant,...*
Stores numeric and string constants to be accessed by a READ statement.
```
1340 DATA NEW YORK, CHICAGO, LOS
     ANGELES, PHILADELPHIA, DETROIT
1350 DATA 2.72, 3.14159, 0.0174533,
     57.29578
```

**DATE$**
Returns today's date.
```
PRINT DATE$
```

**DEFDBL/INT/SNG/STR**
```
DEFDBL  letter,...
DEFINT  letter,...
DEFSNG  letter,...
DEFSTR  letter,...
```
Defines any variables beginning with the *letter(s)* as: (DBL) double precision, (INT) integer, (SNG) single precision, or (STR) string.
```
DEFDBL L-P        DEFINT I-N, W, Z
DEFSNG I, Q-T     DEFSTR A
```

**DEF FN** *function name* [(*argument,....*)] = *function definition*
Defines *function name* according to *function definition*. The *argument* represents those variables in the *function definition* that are to be replaced when the function is called.
```
DEF FNR=RND(90)+9
```

**DEF USR** [*digit*] = *address*
Defines the starting *address* for *digit* assembly-language subroutines.
```
DEF USR3 = &H7D00
DEF USR = (BASE + 16)
```

**DELETE** *line1 - line2*
Deletes from *line1* to *line2* of a program in memory.
```
DELETE 70    DELETE 50-110    DELETE
```

**DIM** *array (dimension(s)), array (dimension(s)),...*
Sets aside storage for the *arrays* with the *dimensions* you specify.
```
DIM AR(100)    DIM L1%(8,25)
```

**EDIT** *line*
  Enters the edit mode so that you can edit *line*.
    EDIT 100   EDIT .

**END**
  Ends execution of a program.
    END

**EOF**(*buffer*)
  Detects the end of a file.
    IF EOF(1) THEN 1540

**ERASE** *array,...*
  Erases one or more *arrays* from a program.
    ERASE C,F

**ERL**
  Returns the line number in which an error has
  occurred.
    PRINT ERL   E = ERL

**ERR**
  Returns the error code (if an error has occurred).
    IF ERR = 7 THEN 1000 ELSE 2000

**ERRS$**
  Returns a system error number and message.
    PRINT "THE LATEST TRSDOS ERROR IS
    ";ERRS$

**ERROR** *code*
  Simulates the error associated with *code* during
  program execution.
    ERROR 1

**EXP** (*number*)
  Calculates the natural exponential of *number*.
    PRINT EXP(-2)

**FIELD** *buffer, length* AS *field name,...*
  Divides a direct-access *buffer* into one or more fields.
  Each field is identified by the *field name* and is the
  *length* you specify.
    FIELD 3, 128 AS A$, 128 AS B$

**FIX** (*number*)
  Returns the truncated integer of *number*.
    PRINT FIX(2.6)

**FOR/NEXT**

FOR *variable* = *initial value* TO *final value* [STEP *increment*]
NEXT [*variable*]
Establishes a program loop.

```
20 FOR H=1 TO -10 STEP -2
30 PRINT H
40 NEXT H
```

**FRE**(*dummy number*) or (*dummy string*)
Returns the amount of free memory space.

```
PRINT FRE(44)
PRINT FRE("44")
```

**GET** *buffer* [*,record number*]
Gets a *record* from a direct disk file and places it in a *buffer*.

```
GET 1      GET 1, 25
```

**GOSUB** *line*
Goes to a subroutine, beginning at the specified *line*.

```
GOSUB 1000
```

**GOTO** *line*
Goes to the specified *line*.

```
GOTO 100
```

**HEX\$** (*number*)
Calculates the hexadecimal value of *number*.

```
PRINT HEX$(30), HEX$(50), HEX$(90)
```

**IF...THEN...ELSE**

IF *expression* THEN *statement(s)* or *line* [ELSE *statement(s)* or *line*]
Tests a conditional expression and makes a decision regarding program flow.

```
10 IF X > 127 THEN PRINT "OUT OF
   RANGE" : END
20 IF A < B THEN PRINT "A < B" ELSE
   IF A = B THEN PRINT "A = B" ELSE
   PRINT "A > B"
```

**INKEY\$**
Returns a keyboard character.

```
A$ = INKEY$
```

**INP**(*port*)
Returns the byte read from a *port*. *Port* may be any integer from 0 to 255.

```
A=INP(42)
```

**INPUT$** (*number* [*,buffer*])
Inputs a string of *number* characters from either the keyboard or a sequential disk file. The *number* must be a value from 1 to 255.
```
A$ = INPUT$(5)    A$ = INPUT$(11,3)
```

**INPUT** [*prompt string;*] *variable1, variable2,...*
Inputs data to a program during execution.
```
INPUT Y%
```

**INPUT#** *buffer, variable,...*
Inputs data from a sequential disk file into one or more *variables.*
```
INPUT#1, A, B    INPUT#4, A$, B$, C$
```

**INSTR**([*integer,*]*string1, string 2*)
Searches for the first occurrence of *string 2* in *string 1* and returns the position at which the match is found.
```
INSTR(A$, "12")
```

**INT**(*number*)
Converts *number* to integer value.
```
PRINT INT(79.89)
```

**KILL** "*filespec*"
Removes *filespec* from the disk.
```
KILL "FILE/BAS"    KILL "DATA:2"
```

**LEFT$**(*string, integer*)
Returns all characters left of position *integer* in the *string.*
```
PRINT LEFT$("BATTLESHIPS", 6)
```

**LEN**(*string*)
Returns the number of characters in *string.*
```
X = LEN(SENTENCE$)
```

**LET** *variable* = *expression*
Assigns the value of *expression* to *variable.*
```
LET A$ = "A ROSE IS A ROSE"
LET B1 = 1.23
```

**LINE INPUT**[*prompt string;*] *string variable*
Inputs a line from the keyboard.
```
LINE INPUT A$
```

**LINE INPUT#** *buffer, variable*
Reads a line of data from a sequential-access file into a string *variable.* The *buffer* is the number used when the file was OPENed.
```
LINE INPUT# 1, A$
```

**LIST** [*startline*] - [*endline*]
Lists program lines to the display.
```
LIST 50     LIST 50-85     LIST -227
```

**LLIST** [*startline*] - [*endline*]
Lists program lines to the line printer.
```
LLIST 780     LLIST 50-     LLIST ,-
```

**LOAD** "*filespec*" [,R]
Loads *filespec*, a BASIC program, into memory. If R is
used, the program is RUN automatically.
```
LOAD "PROG1/BAS:2"
LOAD "PROG1/BAS"
```

**LOC** (*buffer*)
Returns the current record number.
```
IF LOC(1)>55 THEN END
```

**LOF** (*buffer*)
Returns the end-of-file record number.
```
Y = LOF(5)
```

**LOG**(*number*)
Computes the natural logarithm of *number*.
```
PRINT LOG(3.14159)
Z = 10*LOG(PS/P1)
```

**LPOS** (*number*)
Returns the position of the line printer's print head
within the line printer's buffer.
```
IF LPOS(X)>60THEN PRINT CHR$(13)
```

**LPRINT** *data,...*
Prints *data* at the printer.
```
LPRINT (A * 2)/3
```

**LPRINT USING** *format; data,...*
Prints *data* at line printer, using a specified *format*.
```
LPRINT USING "####,#"; 2,17
```

**LSET** *field name* = *data*
Sets *data* in a direct-access buffer *field name*. The
data is left-justified.
```
LSET NM$ = "JIM CRICKET, JR,"
```

**MEM**
Returns the amount of memory.
```
PRINT MEM
```

**MERGE** "*filespec*"
Loads *filespec*, a BASIC program, and merges it with
the program currently in memory.
```
MERGE "PROG2/TXT"
```

**MID$** (*old string, position* [*,length*]) = *replacement
string*
Replaces a portion of *old string* with *replacement
string*.
```
MID$ (A$, 3, 4) = "12345": PRINT A$
```

**MID$** (*string, integer* [*,number*])
Returns a substring of the *string*, beginning with the
*integer* character. *Number* is the number of characters
to include in the substring.
```
MID$ (A$, 3, 2)
```

**MKD$**(*double-precision expression*)
Converts *double-precision expression* to a string
value and returns the 4-byte string.

**MKI$**(*integer expression*)
Converts *integer expression* to a string value and
returns the 8-byte string.

**MKS$**(*single-precision expression*)
Converts *single-precision expression* to a string value
and returns the 2-byte string.

**NAME** *old filespec* AS *new filespec*
Renames *old filespec* as *new filespec*.
```
NAME "FILE" AS "FILE/OLD"
```

**NEW**
Erases a program from memory and clears
all variables.
```
NEW
```

**OCT$**(*number*)
Computes the octal value of *number*.
```
PRINT OCT$(30), OCT$(50),
OCT$(90)
```

**ON ERROR GOTO** *line*
Goes to a subroutine at the *line* specified by the value
of *number*.
```
ON ERROR GOTO 1500
```

**ON** *expression* **GOSUB** *line,...*
Goes to a subroutine at the *line* specified by the value
of *expression*.
```
ON L-1 GOSUB 1000, 2000, 3000
```

**ON** *expression* **GOTO** *line,...*
Goes to the *line* specified by the value of *expression*.
```
ON X GOTO 190, 200, 210
```

**OPEN** *mode, buffer, "filespec"* [,*record length*]
Opens a disk file in the specified mode. (O for
sequential output, I for sequential input, D or R for
direct input/output, and E for sequential extend).
```
OPEN "O", 1, "CLIENTS/TXT"
OPEN "D", 5, "TESTED/BAS", 64
```

**OPTION BASE** *n*
Sets *n* as the minimum value for an array subscript.
```
OPTION BASE 1
```

**OUT** *port, data byte*
Sends *data byte* to a machine output *port*.
```
OUT 32,100
```

**PEEK**(*memory location*)
Returns a byte from *memory location*.
```
A = PEEK (&H5A00)
```

**POKE** *memory location, data byte*
Writes a *data byte* into *memory location*.
```
POKE 15360, 191
```

**POS**(*number*)
Returns the position of the cursor. *Number* is a dummy
argument.
```
PRINT TAB(40) POS(0)
```

**PRINT** *data,...*
Prints numeric or string data on the display.
```
PRINT X,Y PRINT "*"
```

**PRINT TAB**(*n*)
Moves the cursor to the *n* position on the current line
(or on succeeding lines if you specify TAB positions
greater than 79).
```
PRINT TAB(5) "TABBED 5";
TAB(25) "TABBED 25"
```

**PRINT USING** *format; data item,...*
Prints *data items* using the specified *format*.
```
PRINT USING "!";A$;
```

**PRINT** @ *location,*
**PRINT** @ (*row, column*),
Specifies where printing is to begin.
```
PRINT @ 0, "*"
```

**PRINT#** *buffer, item1, item 2,...*
Prints data *items* in a sequential disk file.
```
PRINT#1, A,B
```

**PUT** *buffer* [,*record*]
Puts a *record* in a direct-access file. *Buffer* is the
number used to OPEN the file.
```
PUT 1      PUT 1, 25
```

**RANDOM**
Reseeds the random number generator.
```
RANDOM
```

**READ** *variable,...*
Reads values from a DATA statement and assigns
them to *variables*.
```
READ T    READ S$, T, U
```

**REM**
Inserts a remark line into a program and instructs the
computer to ignore the rest of the program line.
```
10 :REM INPUT SINGLE-PRECISION
20 INPUT A
```

**RENUM** [*new line*] [, *line*] [,*increment*]
Renumbers a program, starting at the specified *line*
and numbering it as *new line*. The optional *increment*
sets the increment to be used between each line
number.
```
RENUM    RENUM 6000, 5000, 100
```

**RESTORE** [*line*]
Restores a program's access to previously read DATA
statements.
```
RESTORE
```

**RESUME** [*line*]
**RESUME NEXT**
Resumes program execution after an error-handling
routine has been performed. RESUME *line* causes
BASIC to branch to the specified line. RESUME NEXT
causes it to branch to the statement following the point
at which the error occurred.
```
RESUME    RESUME 10    RESUME NEXT
```

**RETURN**
Returns control to the line immediately following the
most recently executed GOSUB.
```
RETURN
```

**RIGHT$**(*string, number*)
Returns the last *number* characters of the *string*.
```
PRINT RIGHT$("WATERMELON", 5)
```

**RND**(*number*)
Generates a pseudorandom number between 0 and
the *number*. The *number* must be greater than 0 and
less than 32768.
```
A = RND(2)    A = RND(45)
PRINT RND(0)
```

**ROW** (*number*)
Returns the row position of the cursor. *Number* is a
dummy argument.
```
X = ROW(Y)
```

**RSET** *field name* = *data*
Places *data* in a direct-access buffer *field name*.
```
RSET NM$ = "JIM CRICKET, JR."
```

**RUN** [*line*]
**RUN** *filespec* [,R]
RUN or RUN *line* runs the program that is in memory.
RUN *filespec* loads a program from disk, then runs it.
```
RUN "PROGRAM/A"    RUN "EDITDATA", R
```

**SAVE** "*filespec*" [,A] [,P]
Saves a program in a disk under *filespec*. A causes
the file to be stored in ASCII format. P causes the file
to be stored in an encoded binary format.
```
SAVE "FILE1/BAS,JOHNQDOE:3"
SAVE "MATHPAK/TXT", A
```

**SGN** (*number*)
Determines *number*'s sign. If *number* is positive, SGN
returns 1. If it is negative, SGN returns − 1. If it is zero,
SGN returns 0.
```
Y = SGN(A * B)
```

**SIN**(*number*)
Computes the sine of *number;* the *number* must be in
radians.
```
PRINT SIN(7.96)
```

**SOUND** *tone, duration*
Generates a sound with the specified *tone* (0-7) and
*duration* (0-31).
```
SOUND 1,2
```

**SPACE$**(*number*)
Returns a string of *number* spaces. The *number* must
be from 0 to 255.
```
PRINT "DESCRIPTION" SPACE$(4)
```

**SPC**(*number*)
Prints a line of *number* blanks. The *number* must be
from 0 to 255.
```
PRINT "HELLO" SPC(15) "THERE"
```

**SQR**(*number*)
Calculates the square root of *number*.
```
PRINT SQR(155.7)
```

**STOP**
Stops program execution.
```
STOP
```

**STR$**(*number*)
Converts *number* into a string. If the *number* is positive, STR$ places a blank before the string.

**STRING$**(*number, character*)
Returns a string of the specified *number* of characters. The *number* must be from 0 to 255. The *character* is a string or an ASCII code.
```
B$ = STRING$(25, "X")
PRINT STRING(50, 10)
```

**SWAP** *variable1, variable2*
Exchanges the values of two variables.
```
SWAP F1#, F2#
```

**SYSTEM** [``command''`]
Returns to TRSDOS. If you specify a *command*, TRSDOS executes it and returns you to BASIC.
```
SYSTEM    SYSTEM "DIR"
```

**TAB**(*number*)
Spaces to position *number* on the display. The *number* must be from 0 to 255.
```
PRINT A$ TAB(25) B$
```

**TAN**(*number*)
Computes the tangent of *number*. The *number* must be in radians. If it is in degrees, use TAN (*number* * .11745329). The result is always single precision.
```
PRINT TAN(7.96)
```

**TIME$**
Returns the time (in 24-hour format).
```
A$ = TIME$
```

**TROFF**
Turns off the trace function.
```
TROFF
```

**TRON**
Turns on the trace function (to follow program flow).
```
TRON
```

**USR**[*digit*] (*expression*)
Calls the user's assembly-language subroutine identified by *digit* and passes the result of *expression*.
```
X = USR5(Y)
```

**VAL**(*string*)
Calculates the numeric value of *string*. VAL terminates
its evaluation on the first character that has no
meaning in a numeric term.
```
PRINT VAL("100 DOLLARS")
```

**VARPTR** *variable* or *buffer*
Returns the absolute memory address. When used
with a *variable*, VARPTR returns the address of the
first byte of data identified with *variable*. When used
with a *buffer*, it returns the address of the file's data
buffer.
```
Y = USR1(VARPTR(X))
```

**WAIT** *port, integer1* [*,integer2*]
Suspends program execution until a machine input
*port* develops a specified bit pattern.
```
100 WAIT 32,2
```

**WHILE** *expression*
.
.
.
**WEND**
Executes a series of statements in a loop as long as a
given condition is true.
```
WHILE...WEND
```

**WIDTH** [LPRINT, *integer*]
To set print line width at display or printer. If LPRINT is
omitted, width is set for screen.
```
WIDTH 60
WIDTH LPRINT 132
```

**WRITE** *data,...*
Prints *data* on the display.
```
WRITE A,B,C$
```

**WRITE#** *buffer, data,...*
Writes *data* to a sequential file.
```
WRITE#1, A$,B$
```

# Notes:

# Control Keys

## Command Mode

| | |
|---|---|
| ⟨↑⟩ or ⟨CTRL⟩⟨H⟩ | Backspaces the cursor, erasing the preceding character in the line. |
| ⟨SPACEBAR⟩ | Enters a blank space character and advances the cursor one space. |
| ⟨BREAK⟩ | Interrupts line entry and starts over with a new line. |
| ⟨CTRL⟩⟨J⟩ | Line feed—Starts a new physical lihe without ending the current logical line. |
| ⟨SHIFT⟩⟨0⟩ or ⟨CAPS⟩ | Switches to either all upper case or all lower case. |
| ⟨ENTER⟩ | Ends and enters the current logical line. |

## Execution Mode

| | |
|---|---|
| ⟨SHIFT⟩⟨@⟩ | Pauses execution. Press any other key (except ⟨BREAK⟩) to continue. |
| ⟨BREAK⟩ | Terminates execution and returns to command mode. |
| ⟨ENTER⟩ | Interprets data entered from the keyboard with the INPUT statement. |

# Operators

Each operator or group of operators is precedent over the group below it.

| | |
|---|---|
| ( ) | (Parentheses) |
| $\wedge$ | (Exponentiation) |
| +, − | (Unary negative, positive) |
| *,/ | (Multiplication, division) |
| \ | (Integer division) |
| MOD | (Modulus) |
| +, − | (Addition, subtraction) |
| >,<,=,<=,>=,<> | (Relational tests) |
| NOT | |
| AND | |
| OR | |
| XOR | |
| EQV | |
| IMP | |

# Notes:

# Edit Commands

| | |
|---|---|
| Ⓐ | Moves the cursor to the beginning of the line and cancels editing changes. |
| n(BACKSPACE) | Moves the cursor *n* spaces to the left. If no *n* is given, moves cursor one space to the left. |
| nⒸ | Lets you change *n* characters, beginning at the current cursor position. |
| nⒹ | Deletes *n* characters to the right of the cursor. |
| Ⓔ | Ends editing and saves all changes. |
| (ENTER) | Records all changes and exits edit mode. |
| (SHIFT) (⬆) | Escapes from an insert subcommand (I, H, or X). |
| Ⓗ | Deletes the rest of a line and lets you insert material at the current cursor position. |
| Ⓘ | Lets you insert material at the current cursor position. |
| nⓀc | Deletes all characters up to the *n*th occurrence of character *c* and moves the cursor to that position. |
| Ⓛ | Lists the line. |
| Ⓠ | Quits edit mode and cancels all changes. |
| nⓈc | Searches for *n*th occurrence of character *c* and moves the cursor to that position. |
| n(SPACEBAR) | Moves the cursor *n* spaces to the right. |
| Ⓧ | Displays the rest of the line and lets you add material at the end. |

# Special Characters

| | |
|---|---|
| ' | (apostrophe) Abbreviation for :REM. |
| , | (comma) PRINT punctuation; spaces over to the next 16-column PRINT zone. |
| ; | PRINT punctuation; separates items in a PRINT list but does not add spaces when they are output. |
| : | Separates statements on the same line. |
| . | Indicates current line; use with EDIT and LIST commands. |
| D | Used in double-precision exponential notation. |
| E | Used in single-precision exponential notation. |
| % | Makes variable integer-precision. |
| ! | Makes variable single-precision. |
| # | Makes variable double-precision. |
| $ | Makes variable string type. |

# Notes:

# Error Messages

| Code | Explanation |
| --- | --- |
| 1 | NEXT without FOR |
| 2 | Syntax error |
| 3 | Return without GOSUB |
| 4 | Out of data |
| 5 | Illegal function call |
| 6 | Overflow |
| 7 | Out of memory |
| 8 | Undefined line |
| 9 | Subscript out of range |
| 10 | Redimensioned array |
| 11 | Division by zero |
| 12 | Illegal direct |
| 13 | Type mismatch |
| 14 | Out of string space |
| 15 | String too long |
| 16 | String formula too complex |
| 17 | Can't continue |
| 18 | Undefined user function |
| 19 | No RESUME |
| 20 | RESUME without error |
| 21 | Unprintable error |
| 22 | Missing operand |
| 23 | Line buffer overflow |
| 26 | FOR without NEXT |
| 29 | WHILE without WEND |
| 30 | WEND without WHILE |

## Disk Errors

| Code | Explanation |
| --- | --- |
| 50 | Field overflow |
| 51 | Internal error |
| 52 | Bad file number |
| 53 | File not found |
| 54 | Bad file mode |
| 55 | File already open |
| 57 | Device I/O error |
| 58 | File already exists |
| 61 | Disk full |
| 62 | Input past end |
| 63 | Bad record number |
| 64 | Bad file name |
| 66 | Direct statement in file |
| 67 | Too many files |
| 68 | Disk write protected |
| 69 | File access denied |
| 70 | Command aborted |

## Notes:

# Internal Codes

| Keyword | Code | Keyword | Code |
|---------|------|---------|------|
| ABS | 65414 | INKEY$ | 224 |
| AND | 248 | INP | 65424 |
| ASC | 65429 | INPUT | 133 |
| ATN | 65422 | INSTR | 219 |
| AUTO | 171 | INT | 65413 |
| CALL | 182 | KILL | 200 |
| CDBL | 65438 | LEFT$ | 65409 |
| CHAIN | 185 | LEN | 65426 |
| CHR$ | 65430 | LET | 136 |
| CINT | 65436 | LINE | 177 |
| CLEAR | 146 | LIST | 147 |
| CLOSE | 195 | LLIST | 158 |
| CLS | 159 | LOAD | 196 |
| COMMON | 184 | LOC | 65454 |
| CONT | 153 | LOF | 65455 |
| COS | 65420 | LOG | 65418 |
| CSNG | 65437 | LPOS | 65435 |
| CVD | 65452 | LPRINT | 157 |
| CVI | 65450 | LSET | 201 |
| CVS | 65451 | MEM | 225 |
| DATA | 132 | MERGE | 197 |
| DATE$ | 222 | MID$ | 65411 |
| DEF | 151 | MKD$ | 65458 |
| DEFDBL | 176 | MKI$ | 65456 |
| DEFINT | 174 | MKS$ | 65457 |
| DEFSNG | 175 | MOD | 253 |
| DEFSTR | 173 | NAME | 199 |
| DELETE | 170 | NEW | 148 |
| DIM | 134 | NEXT | 131 |
| EDIT | 167 | NOT | 214 |
| ELSE | 162 | OCT$ | 65433 |
| END | 129 | ON | 149 |
| EOF | 65453 | OPEN | 191 |
| EQV | 251 | OPTION | 186 |
| ERASE | 166 | OR | 249 |
| ERL | 215 | OUT | 156 |
| ERR | 216 | PEEK | 65431 |
| ERROR | 168 | POKE | 152 |
| ERRS$ | 223 | POS | 65425 |
| EXP | 65419 | PRINT | 145 |
| FIELD | 192 | PUT | 194 |
| FIX | 65439 | RANDOM | 187 |
| FN | 212 | READ | 135 |
| FOR | 130 | REM | 143 |
| FRE | 65423 | RENUM | 172 |
| GET | 193 | RESTORE | 140 |
| GOSUB | 141 | RESUME | 169 |
| GOTO | 137 | RETURN | 142 |
| HEX$ | 65434 | RIGHT$ | 65410 |
| IF | 139 | RND | 65416 |
| IMP | 252 | ROW | 65459 |

| | | | |
|---|---|---|---|
| RSET | 202 | TRON | 163 |
| RUN | 138 | USING | 218 |
| SAVE | 203 | USR | 211 |
| SGN | 65412 | VAL | 65428 |
| SIN | 65417 | VARPTR | 221 |
| SOUND | 205 | WAIT | 150 |
| SPACE$ | 65432 | WEND | 181 |
| SPC | 213 | WHILE | 180 |
| SQR | 65415 | WRITE | 183 |
| STEP | 210 | XOR | 250 |
| STOP | 144 | + | 243 |
| STR$ | 65427 | − | 244 |
| STRING$ | 217 | * | 245 |
| SWAP | 165 | / | 246 |
| SYSTEM | 189 | ^ | 247 |
| TAB | 209 | \ | 254 |
| TAN | 65421 | ' | 220 |
| THEN | 208 | > | 240 |
| TIME$ | 226 | = | 241 |
| TO | 207 | < | 242 |
| TROFF | 164 | | |